BINDING A WORKFLOW ENGINE TO A DATA MODEL

FIELD OF THE INVENTION

[0001] This present invention relates generally to workflow management in a computer system and more particularly to binding of a workflow engine to a data model in a computer system.

BACKGROUND OF THE INVENTION

[0002] Typical operation of computerized data processing systems has evolved to find varying solutions to managing the complexity of such systems. Automation has been introduced to relieve operators of repetitive tasks and to increase operator productivity while reducing errors caused by manual actions. The process is managed by computer that assigns work, distributes that work and tracks the progress to completion. One form of such automation has been the introduction of workflows or scripts that are designed to address operational actions and in many cases mimic the responses provided by real operators. Creation and use of such workflows has not been easy as the complex issues involved in managing many such data centre tasks has proved to be very difficult and error prone. Many early examples of computerized workflow were copies of the same manual process used before.

[0003] Complexity appeared to be the root problem to be solved and managing that complexity with workflows or scripts has meant the introduction and use of complex workflows or scripts themselves. In many cases the actual implementation of workflows was overwhelmed by the complexity of the infrastructure. In attempting to replace manual processes with computerized processes no process related changes were typically introduced. New tools were needed to improve the process itself or the management of the process.

[0004] Structured programming techniques have also been applied to address the problem of complex scenario management. Early cases were able to resolve the initial process management issues much in line with the concept of "low hanging fruit". As larger

management issues were faced, dealing with more and more resources and combinations of events, this form of programming proved to be ineffective and too inflexible as it was lacking in capabilities when compared to workflows.

[0005] Therefore what is required is a more effective way to manage the operational complexity of computerized data processing systems.

SUMMARY OF THE INVENTION

[0006] A method, system and program product for binding a workflow engine to a data model representing the real environment is provided. The binding of the workflow engine provides more effective resource selection and flexibility by allowing linking to differing workflows in accordance with the data model. Further results provided by the completed workflow may also be used to augment the data model thereby assuring a more current and consistent representation is found in the data model. Workflow results are therefore used to synchronize the data model with the physical environment modified through workflow requests.

[0007] In one aspect of the present invention, there is provided a method for binding of a workflow engine to a data model containing data objects associated with a plurality of resources for a workflow request having a first message type in a computer system, said method comprising the steps of: updating said workflow request with pre-process workflow data; transforming said updated workflow request from first message type to a second message type supported by said workflow engine; processing said updated workflow request to update said plurality of resources in said computer system; and, updating said data objects of said data model associated with updated said plurality of resources.

[0008] In another aspect of the present invention, there is provided a computer system for binding of a workflow engine to a data model containing data objects associated with a plurality of resources for a workflow request having a first message type in a computer system, said computer system comprising: first updating means to update said workflow request with pre-process workflow data; transforming means to transform said updated

workflow request from first message type to a second message type supported by said workflow engine; processing means to process said updated workflow request to update said plurality of resources in said computer system; and, second updating means to update said data objects of said data model associated with updated said plurality of resources.

[0009] In another aspect of the present invention there is provided a computer program product having a computer readable medium tangibly embodying computer readable program code for instructing a computer to perform the method for binding of a workflow engine to a data model containing data objects associated with a plurality of resources for a workflow request having a first message type in a computer system, said method comprising the steps of: updating said workflow request with a pre-process workflow data; transforming said updated workflow request from first message type to a second message type supported by said workflow engine; processing said updated workflow request to update said plurality of resources in said computer system; and, updating said data objects of said data model associated with updated said plurality of resources.

[0010] In yet another aspect of the present invention there is provided a signal bearing medium having a computer readable signal tangibly embodying computer readable program code for instructing a computer to perform a method for binding of a workflow engine to a data model containing data objects associated with a plurality of resources for a workflow request having a first message type in a computer system, said method comprising the steps of: updating said workflow request with pre-process workflow data; transforming said updated workflow request from first message type to a second message type supported by said workflow engine; processing said updated workflow request to update said plurality of resources in said computer system; and, updating said data objects of said data model associated with updated said plurality of resources.

[0011] In another aspect of the present invention there is provided a computer program product having a computer readable medium tangibly embodying computer readable program code for instructing a computer system to perform the means for binding of a workflow engine to a data model containing data objects associated with a plurality of resources for a workflow request having a first message type in a computer system, said computer system

comprising: first updating means to update said workflow request with pre-process workflow data; transforming means to transform said updated workflow request from first message type to a second message type supported by said workflow engine; processing means to process said updated workflow request to update said plurality of resources in said computer system; and, second updating means to update said data objects of said data model associated with updated said plurality of resources.

[0012] In another aspect of the present invention there is provided a signal bearing medium having a computer readable signal tangibly embodying computer readable program code for instructing a computer to perform the means for binding of a workflow engine to a data model containing data objects associated with a plurality of resources for a workflow request having a first message type in a computer system, said computer system comprising: first updating means to update said workflow request with pre-process workflow data; transforming means to transform said updated workflow request from first message type to a second message type supported by said workflow engine; processing means to process said updated workflow request to update said plurality of resources in said computer system; and, second updating means to update said data objects of said data model associated with updated said plurality of resources.

[0013] Other aspects and features of the present invention will become apparent to those of ordinary skill in the art upon review of the following description of specific embodiments of the invention in conjunction with the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] Preferred embodiments of the invention will now be described, by way of example, with reference to the accompanying drawings, in which:

[0015] FIG.1 is a hardware overview of a computer system, in support of an embodiment of the present invention;

[0016] FIG. 2 is a block diagram of components in an embodiment of the present invention as supported in the computer system of FIG.1;

[0017] FIG. 3 is a flow diagram of determining a workflow in an embodiment of the present invention;

[0018] Like reference numerals refer to corresponding components and steps throughout the drawings. It is to be expressly understood that the description and the drawings are only for the purpose of illustration and as an aid to understanding, and are not intended as a definition of the limits of the invention.

DETAILED DESCRIPTION

[0019] FIG. 1 depicts, in a simplified block diagram, a computer system **100** suitable for implementing embodiments of the present invention. Computer system **100** has processor **110**, which is a programmable processor for executing programmed instructions stored in memory **108**. Memory **108** can also include hard disk, tape or other storage media. While a single CPU is depicted in FIG. 1, it is understood that other forms of computer systems can be used to implement the invention. It is also appreciated that the present invention can be implemented in a distributed computing environment having a plurality of computers communicating via a suitable network **119**.

[0020] CPU **110** is connected to memory **108** either through a dedicated system bus **105** and/or a general system bus **106**. Memory **108** can be a random access semiconductor memory for storing application data for processing such as that in a database partition. Memory **108** is depicted conceptually as a single monolithic entity but it is well known that memory **108** can be arranged in a hierarchy of caches and other memory devices. FIG. 1 illustrates that operating system **120** may reside in memory **108.**

[0021] Operating system **120** provides functions such as device interfaces, memory management, multiple task management, and the like as known in the art. CPU **110** can be suitably programmed to read, load, and execute instructions of operating system **120**. Computer system **100** has the necessary subsystems and functional components to implement selective program tracing functions such as gathering trace records and historical data as will be discussed later. Other programs (not shown) include server software applications in which

network adapter **118** interacts with the server software application to enable computer system **100** to function as a network server via network **119**.

[0022] General system bus **106** supports transfer of data, commands, and other information between various subsystems of computer system **100**. While shown in simplified form as a single bus, bus **106** can be structured as multiple buses arranged in hierarchical form. Display adapter **114** supports video display device **115**, which is a cathode-ray tube display or a display based upon other suitable display technology. The Input/output adapter **112** supports devices suited for input and output, such as keyboard or mouse device **113**, and a disk drive unit (not shown). Storage adapter **142** supports one or more data storage devices **144**, which could include a magnetic hard disk drive or CD-ROM, although other types of data storage devices can be used, including removable media.

[0023] Adapter **117** is used for operationally connecting many types of peripheral computing devices to computer system **100** via bus **106**, such as printers, bus adapters, and other computers using one or more protocols including Token Ring, LAN connections, as known in the art. Network adapter **118** provides a physical interface to a suitable network **119**, such as the Internet. Network adapter **118** includes a modem that can be connected to a telephone line for accessing network **119**. Computer system **100** can be connected to another network server via a local area network using an appropriate network protocol and the network server that can in turn be connected to the Internet. FIG. **1** is intended as an exemplary representation of computer system **100** by which embodiments of the present invention can be implemented. It is understood that in other computer systems, many variations in system configuration are possible in addition to those mentioned here.

[0024] FIG. **2** is a simplified view of the logical relationship between the software components of an embodiment of the present invention. Receiving front end **200** may be advantageously implemented as a SOAP style interface within a web services component allowing a broad range of access to requesters of workflow. Other means of implementing such an access point may be used as well as is known in the art. Pre-workflow interceptor **210** catches incoming workflow requests to allow pre-processing to be performed as needed. Pre-processing includes substitution of workflow data by data from other sources such as

data objects held in data model **270**. Interceptor **210** may be invoked externally through request from other components or internally by way of the executing workflow as it initiates a sub-workflow which is another workflow itself. Message translator **220** provides transformation services between various supported message formats. The message output is used to actually commence a workflow. Queue **230** is used to contain workflows scheduled for execution providing a staging or holding place. Deployment engine **240** performs the actual workflow execution. Results of the workflow being run in deployment engine **240** are provided in the form of results message **250**. Post-workflow interceptor **260** catches the resulting messages or outcome notification from deployment engine **240** processing to allow for any additional processing to occur. Finally data model **270** is updated with results of processing workflow by post-workflow interceptor **260** to include any updates specified in the logical operations of the workflow. The updating performed keeps the data model synchronized with actual real infrastructure view **280**. If the workflow processing was unsuccessful, then post-workflow interceptor **260** would simply perform cleanup activities to remove any partially updated data objects as necessary. Pre-workflow interceptor **210** could be used to disable monitoring of devices scheduled to be updated by a workflow. In this case the execution of the workflow would not cause any unnecessary alerts as monitoring for those specific devices would have been disabled. Similarly post-workflow interceptor **260** would have been use to enable monitoring of the devices, for which monitoring was disabled, after completion of the workflow execution. Real infrastructure view **280** is the actual view of the infrastructure associated with the complex being managed. Real infrastructure view **280** may contain data for hardware and software implementations.

[0025] Synchronization of real infrastructure view **280** and data model **270** is performed routinely by post-workflow interceptor **260** removing the burden of this activity from the actual workflow. Having synchronized real infrastructure view **280** and data model **270,** workflows can now leverage data contained in data model **270** through pre-workflow interceptor **210** providing added simplification for workflow designers.

[0026] Deployment engine **240** has been enabled to process and invoke logical operations contained within workflows. This enablement allows workflows to be further adapted to data model **270**. As a result, workflows may be structured to include multiple

logical operations, each of which can invoke a different sub-workflow depending upon data model **270**. This then mimics the object-oriented approach toward workflow development. Further the late binding to the workflow of information in data model **270** allows the workflow to link to other workflows in accordance with the data in data model **270.**

[0027] In an example implementation of an embodiment of the present invention using a Java environment, enterprise Java beans may be used to comprise the interceptor layer. A stateless session bean may then be used for pre-workflow interceptor **210** with a message driven bean being used for post-workflow interceptor **260**. The entry point for receiving requests through a SOAP interface may be implemented as the stateless session bean (as in pre-workflow interceptor **210**) and be exposed over web services description language (WSDL) to appeal to a broad array of potential users.

[0028] FIG. **3** is a flow diagram of an embodiment of the present invention showing detail of pre-workflow interceptor **210** of FIG.**2** determining a workflow. Pre-workflow interceptor **210** of FIG.**2** is used in the example to resolve a workflow prior to execution. During operation **300** necessary data centre model objects are made available from a metadata repository (the data centre model) to be used throughout further processing by pre-workflow interceptor **210**. The metadata repository contains a plurality of identified objects describing or representing the data centre resources to be managed by the associated workflows. The tertiary association of metadata objects corresponding to logical operations with various data centre model objects identified and their device models and workflows, for example, then allow resolution of any logical operation with a data centre model identifier to a workflow. The general flow is to traverse a search hierarchy of workflow requests, device models and defaults as described in the following.

[0029] During operation **310** a determination is made with regards to the workflow and associated request types. Request types are the logical operations contained within the workflow being requested. These are enumerated and provide a quick way to indicate what resources and actions are intended for use by the workflow. If the requests types are found, the workflow is ready for execution and the flow moves to operation **320** where the workflow is then executed to completion. Otherwise, having not found request types,

processing moves to operation **330**.

[0030] During operation **330** device model **270** is queried for appropriate device information to be used by the workflow. If successful, processing moves to operation **340** where the workflow is again queried for request types. If this query is satisfied, processing then moves to operation **320** where the workflow is executed as before. The workflow is queried to determine if changes have been made due to late binding with the data model to satisfy information related to request types or other pre-workflow processing that may have occurred. If on the other hand, lookup in device model **270** did not provide a favourable response, processing would move to operation **350**.

[0031] During operation **350** a determination is made with regard to default specifications for processing of the requested workflow. If default is specified then processing moves to operation **360** during which a default device specification will be examined to determine if it meets the needs of the workflow. If the default device specification can satisfy the workflow request, then processing moves to operation **370** where again the requested workflow is queried for appropriate request type information. If the request type information is available in keeping with the default device specifications then processing moves to operation **320** where the workflow is then executed as before.

[0032] If the determination made during operation **350** resulted in no defaults specified the workflow would have been stopped as the processing would move to end at operation **380**, during which any necessary cleanup of resources would have been performed. Similarly if during operation **360** no default device could be selected or made available, then processing would have moved to end at operation **380**. Also if during operation **370** no request type information could be obtained in conjunction with a default device specification, then that result would cause processing to move to end at operation **380** as before.

[0033] Having successfully identified the workflow, the workflow may be executed during operation **320** by calling the workflow engine shown as deployment engine **240** of **FIG. 2**.

[0034] Pre-processing of the workflow eliminates the need to manage conditions during

the workflow itself. For example, a workflow may require the operation of a server at a specific IP address. Prior to execution pre-workflow interceptor **210** would intercept the workflow request, and act upon the conditional process requirement by delegating to a typical availability monitor to determine if the server is available. If the server was not available, the workflow would not be executed and the request would fail. Otherwise, knowing the server was available the workflow would be safely executed without server availability concerns. No extra checking is required during the workflow processing due to the separation of the task from the workflow. This separation makes the creation of workflows simpler

[0035] Although the invention has been described with reference to illustrative embodiments, it is to be understood that the invention is not limited to these precise embodiments and that various changes and modifications may be effected therein by one skilled in the art. All such changes and modifications are intended to be encompassed in the appended claims.